EFFICIENT INTEGRATION OF OLD AND NEW RESEARCH TOOLS FOR AUTOMATING THE IDENTIFICATION AND ANALYSIS OF SEISMIC REFERENCE EVENTS

D. Wilmer Rivers,[1] Craig A. Schultz,[2] and Douglas A. Dodge[2]

Multimax, Inc.,[1] Lawrence Livermore National Laboratory[2]

Sponsored by National Nuclear Security Administration
Office of Nonproliferation Research and Engineering
Office of Defense Nuclear Nonproliferation

Contract No. DE-FG02-01ER83218[1] and W-7405-ENG-48[2]

ABSTRACT

The selection and study of reference events for inclusion in the National Nuclear Security Administration (NNSA) Knowledge Base requires the application of a much broader suite of seismic analysis software than does either the routine production of a seismic bulletin or the subsequent preliminary screening of those bulletin events to conduct nuclear explosion monitoring. For either of those latter applications, a large program designed explicitly for that single purpose can be applied in a set analysis procedure for every one of the many events that will be processed daily, but to study the much smaller number of reference events in adequate detail, it is necessary to use many separate specialized programs, including new routines that may still be in the testing phase or even some that were developed or modified specifically to study the particular event at hand. In order to make the most effective use of this software suite, the individual programs must communicate with one another in an efficient and flexible manner, so that a scientist can select which programs to run and in what order, thereby improvising a data-processing pipeline of programs passing their results from one to the next.

We have explored the technical feasibility of constructing such a reference event analysis system by using the Common Object Request Broker Architecture (CORBA) as the data communications tool for passing data among programs and for allowing them to make Remote Procedure Calls (RPCs) to invoke one another's services. We have extended the standard seismic analysis program *geotool* to incorporate a CORBA interface, and through it, we can send data and RPC messages to other programs for processing, and then back again to *geotool* for interactive graphics display of the results. The communications take place through separate Object Request Brokers (ORBs) for *geotool* and for the analysis programs. We demonstrate this procedure by using the C-language program *geotool* as server software on Linux and a Java application for waveform filtering as client software in MS Windows. The two programs on the separate platforms are able to function together just as if the remote Java application were actually a local C-language routine within *geotool* called via a pull-down menu. All data communications take place within memory, and no new disk files or database records need be generated as output from one program and then read as input to the other. It is this type of seamless data communications that should form the infrastructure for integrating the many separate programs that will be needed to analyze seismic reference events thoroughly.

Although the work performed to date has used CORBA as the mechanism for inter-process communications, we have monitored the software industry's increasing acceptance within the last year of "Web services" as an alternative approach to the integration of distributed software components. Web services technology is based on Simple Object Access Protocol (SOAP) invocations of remote procedures, and the SOAP calls are transmitted through XML-formatted text messages sent via HTTP. Since the messages are text-based, and since they are handled by a SOAP-enabled Web server rather than by an ORB, inter-platform data communications are simpler (but more bandwidth intensive) through Web services than through CORBA. The potential for integrating native Windows applications with UNIX applications through Web services is particularly appealing, so in our future work, we intend to investigate the use of Web services as an alternative to CORBA for transmitting data and RPCs among the programs that will be used to analyze reference events. In particular, the XML messages invoking the remote processing offer a means for creating a metadata tracking system that can be used to record the data-processing history of each parameter measurement made by the distributed system of independent software components, so the metadata for the entire suite of data analysis procedures for each waveform can be archived for repeating (or undoing) any or all of the measurements.

OBJECTIVE

Scientists use a variety of stand-alone computer programs to analyze and identify seismic events for nuclear explosion and treaty monitoring, and an especially wide selection of software tools is needed for the detection and intensive study of reference events that will be included in the NNSA Knowledge Base for use in future event comparisons. Some of these stand-alone programs are large software packages that offer many tools for routine seismic analysis, but they are difficult to modify to include additional tools for specialized tasks. Others of these stand-alone programs offer the capability of performing only specific operations, and they must be used in conjunction with other programs such as interactive waveform graphics displays that offer more general analysis capabilities. In most cases neither the large software packages nor the specialized analysis programs can communicate adequately from one to another without the tedious creation and input of temporary data files and other awkward techniques. Since the stand-alone programs cannot exchange data easily, it is difficult to use them in a data-processing pipeline that could add new capabilities to those offered by the large software packages or that could allow the specialized analysis programs to rely on other software for tasks such as graphics displays.

A reference event analysis system should therefore be built by using a system architecture that facilitates data flow among these stand-alone programs, including any new programs that will be developed in the course of future research and that may be especially valuable for the identification and characterization of reference events. This new architecture should allow the results of one program to be sent easily to another one, as chosen on a case-by-case basis by the seismic analyst, without the creation of temporary files and database tables. Because many of the stand-alone seismic analysis programs that need to communicate with one another are written in different computer languages, and many are written for use under different operating systems, it will be important for this architecture to be as nearly platform-independent as possible. Furthermore, the communications among the separate programs should allow access to remote resources for data retrieval or specialized computations. The reference event analysis system should therefore be constructed as a distributed system of individual software components rather than as a single large software package. The first objective of our study is to examine software architectures and communications protocols that will allow existing and newly developed programs to be used as the components in this distributed system. Our next objective is to select and modify those programs so that they can be integrated into a reference event analysis system using that distributed system architecture.

RESEARCH ACCOMPLISHED

This project is divided into two phases: a six-month proof-of-concept phase, which has recently been completed, and a two-year implementation phase, which has now begun. The proof of concept was intended to explore the architectural underpinnings of a software system for analyzing seismic reference events, and it was not intended in this first stage to design and construct any of the graphics, signal processing, geophysical, communications and other modules that would be the constituent parts of such a system. Our study has not focused so much on the construction of a new seismic analysis package that is intended to replace the ones currently in use as it has focused on the integration of separate modules, including both existing code and newly developed routines, into a flexible software system. We shall select the individual modules to use for analyzing reference events, and we shall build certain new data processing tools as required, during the two-year implementation phase that is now underway.

In examining possible architectures for a reference event analysis system, our choices have been guided strongly by the differences between routine seismic analysis performed for constructing event bulletins and the more nearly *ad hoc* analysis that is required to identify and characterize reference events for inclusion in the Knowledge Base. Software packages for routine seismic analysis are designed to facilitate the repetitive performance of a pre-determined series of tasks. Although a data analyst engaged in constructing an event bulletin will have some flexibility in choosing to perform certain ones of those analysis tasks for every waveform under consideration while reserving other analysis tasks for use only in unusual circumstances (such as using a polarization filter to help separate the sources of mixed signal arrivals), by and large the processing of every new event will conform to set procedures. In particular, if the analyst makes any choice of data processing tools at all, the selection of those tools will be limited to only those modules that are included within the seismic analysis software package, so the designer of that package must therefore foresee all the tasks that the analyst may need to perform. For screening all seismograms recorded by a network and pre-processed by an automated system, and especially for making a specified set of measurements that are agreed upon for international data exchange, this approach is, in fact, a suitable one. A number of software packages such as Seismic Analysis Code (SAC2000; *viz.* Goldstein, 1998),

Analyst Review Station (ARS; *viz.* Wang, 1996), *geotool* (Henson, 1993), and MatSeis (Young, 2001), among others, are commonly used for these analysis tasks. These same packages can also be used satisfactorily for the analysis of reference events, but we feel that the use of any one of these packages, or for that matter the use of any other single large program that we could design and build *ab initio* as a product of our current study, may not be the best possible approach for this purpose. We shall now discuss some shortcomings of that approach and describe alternatives to it.

For identifying and characterizing possible reference events for use within the Knowledge Base, it is critical that the scientist have maximum flexibility in choosing what algorithms and data processing routines should be used in any particular case. The purpose of a reference event is to exemplify the seismograms that are to be expected from a particular type of event, in a particular region, as recorded at particular stations, and the tools needed to characterize those seismograms will be as varied as the waveforms themselves. A wavetrain from a small strike-slip earthquake in the upper mantle in a region of low anelastic attenuation will, of course, be markedly different from the waveform of a large thrust-fault earthquake at shallow depth in a tectonically active region, and a scientist may wish to choose to apply a different suite of tools depending on certain features that are exhibited by one of these events but not by the other. In fact, highlighting these differences in order to categorize patterns that can be used to distinguish one type of event from another is one of the goals of reference event analysis. Even for conventional applications such as picking arrival times, determining epicenters and confidence regions, measuring signal amplitudes and periods, etc., when studying a reference event, a scientist may well wish to apply unusual tools that would not conventionally be applied during routine event screening. For instance, elaborate de-ghosting algorithms may be used to identify multiple small echoes within reference signals propagating along paths characterized by strong multipath arrivals, a procedure that is unlikely to be applied to signals not intended for inclusion in the Knowledge Base. It is even possible that a scientist may wish to develop special software for application to a particular signal, if no tool that is available seems appropriate for the purpose. Applying specialized tools would require making modifications or extensions to the seismic analysis software packages. Another reason that these packages could need to be modified for reference event analysis is that it can be the software itself, rather than the seismogram, that is the subject of the analysis. This would happen because a scientist who is developing a new analysis routine will likely want to test it by applying it to reference signals in conjunction with the existing tools that already exist in those packages. For instance, a scientist who has developed some variant on the conventional computation of the complex signal cepstrum will want to test only that algorithm and not have to write new graphics code for the display of the waveforms, spectra, cepstra, etc., capabilities that are already offered by the existing seismic analysis packages. For processing reference events, then, it will be necessary for a scientist to make modifications to SAC, ARS, *geotool*, etc., by incorporating unusual or newly developed analysis tools into the software package.

Although most of these seismic analysis software packages can be modified to some extent, the process is usually not an easy one. In some cases modifications can be made to the software through the resource files it processes as data, and in other cases new libraries can be linked in and operated from existing callbacks, but often it is necessary to modify (and thus have access to) the source code to make some desired changes. Even when it is possible to do this, it is dangerous to do so, since modifying code in the huge single programs that form the basis of most of these packages is liable to result in unanticipated "side effects" that cause existing functionality in the program to break or to behave in a different manner than before. Another major problem is that often a scientist who is working with a large seismic analysis program written in the C programming language, for example, will want to implement a processing routine that is written in Java or some other language. Although software programs can be constructed to invoke routines written in one language from main programs written in another, the process is not so straightforward that a scientist or data analyst could rig it together in short order as an improvised addition to the analysis of a specific event. Scientists who are running a program like *geotool* that is intrinsically hard coded with calls to a particular graphics platform (in this case, Motif) are furthermore constrained to work within a particular operating system such as UNIX and therefore cannot take advantage of seismic analysis codes written to run on another platform such as Windows. To overcome these problems associated with adding functionality to large software systems for seismic analysis, we have decided to take a different approach to constructing the reference event analysis system. Instead of augmenting an existing large program, or writing a new one that would be more nearly comprehensive but that would itself eventually become inadequate when future research results in new analysis algorithms and new software tools, we have chosen to implement a distributed architecture that will permit existing code as well as new code to be invoked remotely from the large programs currently used for seismic analysis.

The distributed system architecture that we have investigated during the recently concluded proof-of-concept phase is based on CORBA, a technology that was developed in the mid-1990s for client/server data communications between desktop computers and corporate mainframes. Our scheme for implementing CORBA as the backbone for a distributed seismic analysis software system is illustrated in Figure 1. We are using the C-language program *geotool* as the server software, running under the Linux operating system, and for the proof of concept, we have used a simple Java program called "WaveformViewer" as a client application, running under the Windows 2000 operating system on a separate computer within the same local area network (LAN). As its name implies, WaveformViewer does little more than display waveforms using Java graphics, but it does offer a limited functionality, such as digital filtering, that will suffice to demonstrate how the reference event analysis system should work. WaveformViewer is, in fact, assembled from only a few of the many Java classes that make up a much more comprehensive seismic analysis system (Henson *et al.*, 2000), but rather than use that full system, thereby linking together two large programs that offer many duplicate capabilities, we wish to show how a single server can drive data analysis by communicating with a number of small client programs, each of which perform only a small number of particular tasks, perhaps only a single function. This is the system design that we intend to implement in the reference event analysis system.

As is shown in the diagram in Figure 1, we have modified *geotool* by adding new callback functions that allow it to communicate with an ORB. (Since *geotool* is running under Red Hat's distribution of Linux, we can make use of the open-source ORB known as "ORBit" that is bundled as a part of the distribution. For a *geotool* server running instead under Solaris, we could use one of a number of commercial ORBs for UNIX.) Using the X Resources file for *geotool*, we can then add a new pull-down menu item that will allow the user to send data to our client program, WaveformViewer. To expedite the process (external to *geotoo*l) of setting up the data connection, we have written a utility called "CORBA Center" that effectively acts as a switchboard for choosing a server side and a client side of the data flow. This utility registers the IP addresses with the CORBA Naming Service (to associate object references with symbolic names) so that the two programs can exchange data across the LAN as easily as if they were both running on the same computer. CORBA's data communication across the LAN takes place using the Internet Inter-ORB Protocol (IIOP). This is an official internet protocol (like FTP and HTTP) that allows an ORB on one platform to communicate with one on another platform. Note that it is irrelevant that the server-side software is in C and the client-side software is in Java, since IIOP is platform independent. We have modified the WaveformViewer client application by adding callback functions that allow it to communicate with a new "Agent" Java class, which, in turn, handles the communication with the ORB that is included within the Java platform under release JDK 1.2. The data messages that are exchanged through this client/server communication consist of seismograms and the metadata that describe them, in conformance with the standard CSS data schema (Carter *et al.*, 2001). To allow this communication, we have translated the CSS-schema data structures such as *.wfdisc*, *.arrival*, *.origin*, etc., into CORBA's Interface Definition Language (IDL). An IDL routine does not know whether the application with which it is communicating is written in C or Java (or one of several other languages). All it knows is that the application on the other side of the data stream expects to receive a message conforming to a particular IDL argument list, and we have therefore translated all the standard CSS data tables into a single IDL interface so that we can re-use that same interface for as many different *geotool* client/server applications as possible.

Figure 2 shows a server-side view of how this system can be used in practice. Once the program "CORBA Center" has been run to specify that *geotool* on the Linux computer will be a server and that WaveformViewer on the Windows computer will be a client, the data analyst starts up *geotool* in the normal manner. To make it readily apparent on the screen what is happening within the client/server data flow, in Figure 1 the analyst has used *geotool*'s own data-processing functionality to apply a high-pass filter to two seismograms. Those two seismograms are thus now different within *geotool* from the versions that were read in from the disk file, and (depending on what processing was applied) perhaps they are no longer correctly described by the CSS-schema tables residing in the database. The analyst will now use a new menu item that we have added to *geotool* to send to the client whichever seismograms are selected on the screen. In this case we shall send to WaveformViewer five seismograms, including the two that we have changed within *geotool*. These waveforms will be sent to the client directly from memory, so the analyst will not have to write the new waveforms to disk, then store new CSS-schema tables in the database, and then send those new disk-resident data to the Windows computer via FTP, where they would next have to be written to that computer's disk and finally read in by WaveformViewer. Instead, the data are sent directly between the programs through the IDL interfaces. This is in effect a client/server RPC, but it is considerably easier to implement using the high-level CORBA architecture than it would be by using a low-level interface such as code for sockets.

Figure 3 is a screen shot from the Java program WaveformViewer running on the Windows platform, showing a callback we have added that displays the *.wfdisc* description of the waveforms received through the IDL interface, just as if they had been read from the disk. The analyst can select which seismograms to display within the Java-graphics WaveformViewer display (which mimics the Motif waveform display used by *geotool*), and in this case we choose to display all five waveforms. In Figure 5 the analyst has applied WaveformViewer's own filtering routine to apply a different high-pass filter to two of the seismograms that were received from geotool as unfiltered waveforms. The analyst will then use a menu item we have added to WaveformViewer to send data back to the server program. Figure 5 shows that the waveforms newly modified by WaveformViewer are indeed now present within *geotool*. The whole process is as simple for the analyst to use as it would be if all the digital filtering had been performed within *geotool* itself. All that is required is to make a few mouse clicks, just as if WaveformViewer were simply a function that is part of *geotool* instead of its being in fact a separate program, written in a different language, running on a different computer and under a different operating system.

This proof of concept demonstrates that a distributed software architecture built using CORBA can form the basis of a reference event analysis system. Instead of having a single huge program, which is hard and/or dangerous to modify, as the tool to use for identifying and characterizing reference events, it is possible to modify *geotool* (or ARS, or SAC, etc.) to be used as a data server and then to perform the actual seismic analysis externally to *geotool* by using a host of separate applications, including both existing code and programs yet to be written, running on whatever platform is most appropriate. In particular, the analyst should be able to use seismic data centers and supercomputers located at remote facilities accessible via the Internet, just as if those capabilities were offered by the server platform itself. The data communications should be performed as easily as possible without requiring the analyst to go through intermediate steps of creating, exporting, and importing temporary files and database records. This flexibility will be required for the intensive study of reference events, where new algorithms not ordinarily used for routine seismic analysis may be required in order to perform specialized investigations in individual cases. The analyst should be able to select which programs to run, and in what order to run them, and in many cases it may well be necessary to construct new programs. The analysis programs that run as client applications should perform only one or two specific tasks, and they can be written to perform only those particular functions without reproducing all the overhead capabilities for data management that are required by the server program.

The example that is illustrated in Figures 2 – 5 should serve to demonstrate the need for certain features in a distributed processing architecture that are not conventionally part of a software system like *geotool* that runs as a single program on one computer (albeit within multiple windows). In Figure 2 data channels 6 and 8 are unfiltered, in Figure 4 they have been filtered by the client application, and in Figure 6 they have been returned to the *geotool* server in their filtered form. What, then, should the *geotool* server program do with the unfiltered versions of these channels, which, of course, are still resident in the program's RAM, and are still displayed on the Linux computer screen, once the client program is ready to send the filtered versions back to the server? Should the process of returning the filtered data from the client automatically cause the server to delete the unfiltered data? Should the *geotool* window instead now display the presence of two different versions of these waveforms, and if so, should it change the waveform identification label *.wfid* to reflect that there are now two different waveforms for the same time windows on these particular data channels? Should the return of the filtered data from the server be blocked until the analyst has hit a "receive" button in *geotool* indicating that the original data should now be overwritten? (Blocking the transmission of a data message from the client until the server specifies that it is ready to receive it makes the data stream function in many ways like an Instant Messenger service such as AOL IM or ICQ.) What happens if the data analyst sends the unfiltered data to one client application for filtering and then chooses to send those same data to a different client program for some other processing, such as polarity reversal, *before* the first client program has returned its results? Clearly, the distributed processing architecture presents a number of issues related to the data's referential integrity, versioning, and configuration management. Establishing and enforcing policies for handling these issues will be an important part of the design of the reference event analysis system.

It is also clear that the system must use some sort of metadata audit trail for tracking the distributed processing. Regardless of whether the unfiltered data are deleted from the server's memory and screen display when the filtered data are returned from the client, how will the system distinguish between the filtered waveforms and the original ones that reside on the disk? In the particular case of the interactive WaveformViewer client application, the analyst was required to input the filter parameters by using a GUI, so it is known (at least by the analyst, if not by *geotool*) what sort of filter was applied. In general, however, it is far more likely that the client code will run automatically, and the analyst will have no knowledge of the exact parameters that govern the algorithm implemented by the client

program, which, after all, is likely to be running on a different computer, perhaps even at a different facility. The client must therefore return not only the processed data but also a metadata message describing just what operations it performed and the parameters it used. The server must construct a database archive of these metadata records as a tool to use for the data versioning and configuration management so that any one or all of the data processing routines executed by the distributed client applications can later be repeated or undone.

Finally, although our work has shown that CORBA can be used as the backbone for the reference event analysis system, we have now begun an investigation into the use of an alternative technology. Within the last year the software industry has begun to move from using CORBA for client/server applications to an Internet- or Intranet-hosted approach based on Web services. This technology relies on Web servers, rather than ORBs, for data communications, and the data (including SOAP's RPC invocations to process the data) are sent as XML messages via HTTP rather than as binary data sent via IIOP. There are a number of advantages to this approach. Because XML messages are alphanumeric data, they can be sent through firewalls more easily than can binary data (although more bandwidth is required). XML and SOAP are easier to use than CORBA (even though we have made that process easier by the construction of the "CORBA Center" program that facilitated the *geotool*-to-WaveformViewer connection). Because XML is just alphanumeric data, it is truly platform independent. In particular, it is quite difficult to use CORBA to communicate with applications that are running under Windows as native code (as opposed to applications like WaveformViewer that run under Windows but within Java's own platform), since the standard Windows platform uses the Component Object Model (COM), which is incompatible with the CORBA object model. Software bridges can be constructed, but the process is difficult. XML bypasses this problem, and so Web services that are developed for UNIX work perfectly with those that are developed for Windows. Microsoft has made the construction of Web services especially easy by now replacing COM with a new platform called ".NET" that automates much of the use of SOAP. We feel that this platform will be an important one, and the reference event analysis system should be able to make use of it. Using XML as the basis for data communications has further advantages, including the fact that in a real sense, XML is itself metadata. Database vendors are therefore making sure that new releases of their software can store and transmit XML. We feel that this would be a good approach for constructing the metadata archive that will be an important component of the planned system.

CONCLUSIONS AND RECOMMENDATIONS

We conclude that the best system architecture for a reference event analysis system is likely to be one that is based on distributed stand-alone software components (which can all be on the same computer or distributed across a network) rather than on a single large program. Our proof of concept study has shown that CORBA technology can be used to connect a server program for seismic analysis (like *geotool*) to client programs developed for specialized applications, even if they are written in other languages and run under different operating systems. We recommend, however, that Web services technology be investigated thoroughly as an alternative to the use of CORBA as the backbone for the distributed processing. We shall implement that recommendation during the next phase of work, and then we shall begin incorporating both existing and new programs into that distributed processing system

REFERENCES

Carter, J., R. Bowman, K. Biegalski, J. Bohlin, M. Fisk, R. Carlson, W. Farrell, B. MacRitchie, and H. Magyar (2001), IDC Documentation Database Schema Revision 3, Center for Monitoring Research User Guide on-line document (available at http://www.pidc.org/librarybox/idcdocs/downloads/511r3ab.pdf).

Goldstein, P. (1998), SAC Tutorial Guide for New Users, Lawrence Livermore National Laboratory report UCRL-MA-112836 (available at http://www.llnl.gov/sac/).

Henson, I. (1993), The *geotool* Seismic Analysis System, in: Proceedings of the 15th Annual Seismic Research Symposium 8-10 September 1993, Phillips Laboratory report PL-TR-93-2160.

Henson, I., R. Wagner, and W. Rivers (2000), Automated Seismic Event Monitoring System, U.S. Nuclear Regulatory Commission report NUREG/CR-6625.

Wang, J. (1996), Analyst Review Station's User Manual, Science Applications International Corporation report SAIC-96/1100.

Young, C. (2001), MatSeis User's Manual, Version 1.3, Sandia National Laboratories on-line document (available at http://www.nemre.nn.doe.gov/nemre/data/matseis/matseis_manual.pdf).

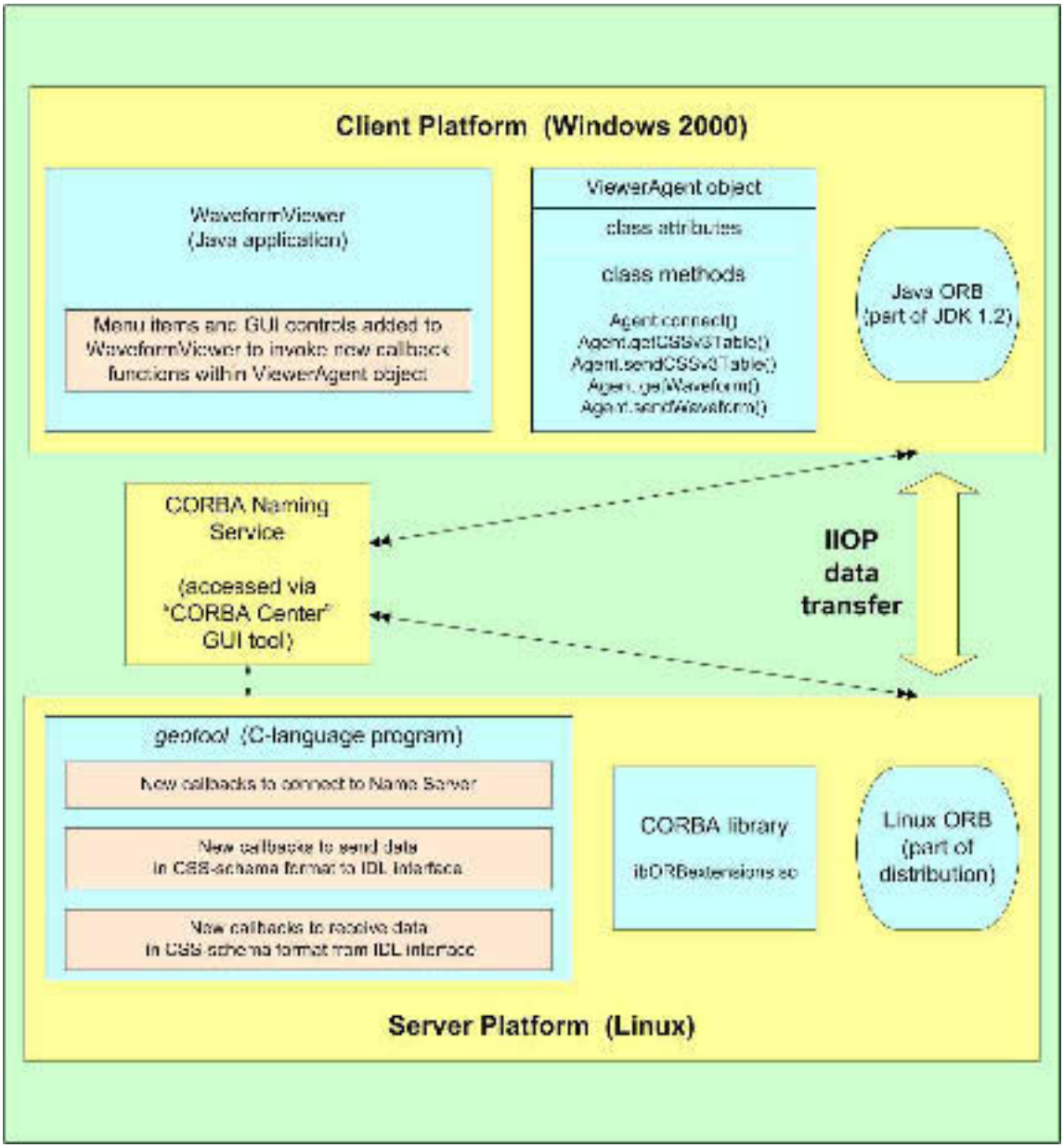

Figure 1. Using CORBA to enable client/server data communications between the C-language program *geotool*, running as a server on a Linux platform, and the Java program WaveformViewer, running as a client application on a Windows 2000 platform. The data communication takes place between the Object Request Brokers on each platform via the Internet Inter-ORB Protocol, which is an internet standard. We have modified the client code and the server code so that they link to CORBA, and that link can now be used to permit data communications between the sever and additional client applications that are invoked by the user through items added to the *geotool* pull-down menus. The data that are exchanged between the client and server programs are translated from the standard CSS database table schema to CORBA Interface Definition Language. We have written a utility called "CORBA Center" that makes it easier for the user to set up the client/server link (which requires the *geotoo*l server to register itself with the CORBA Name Server).

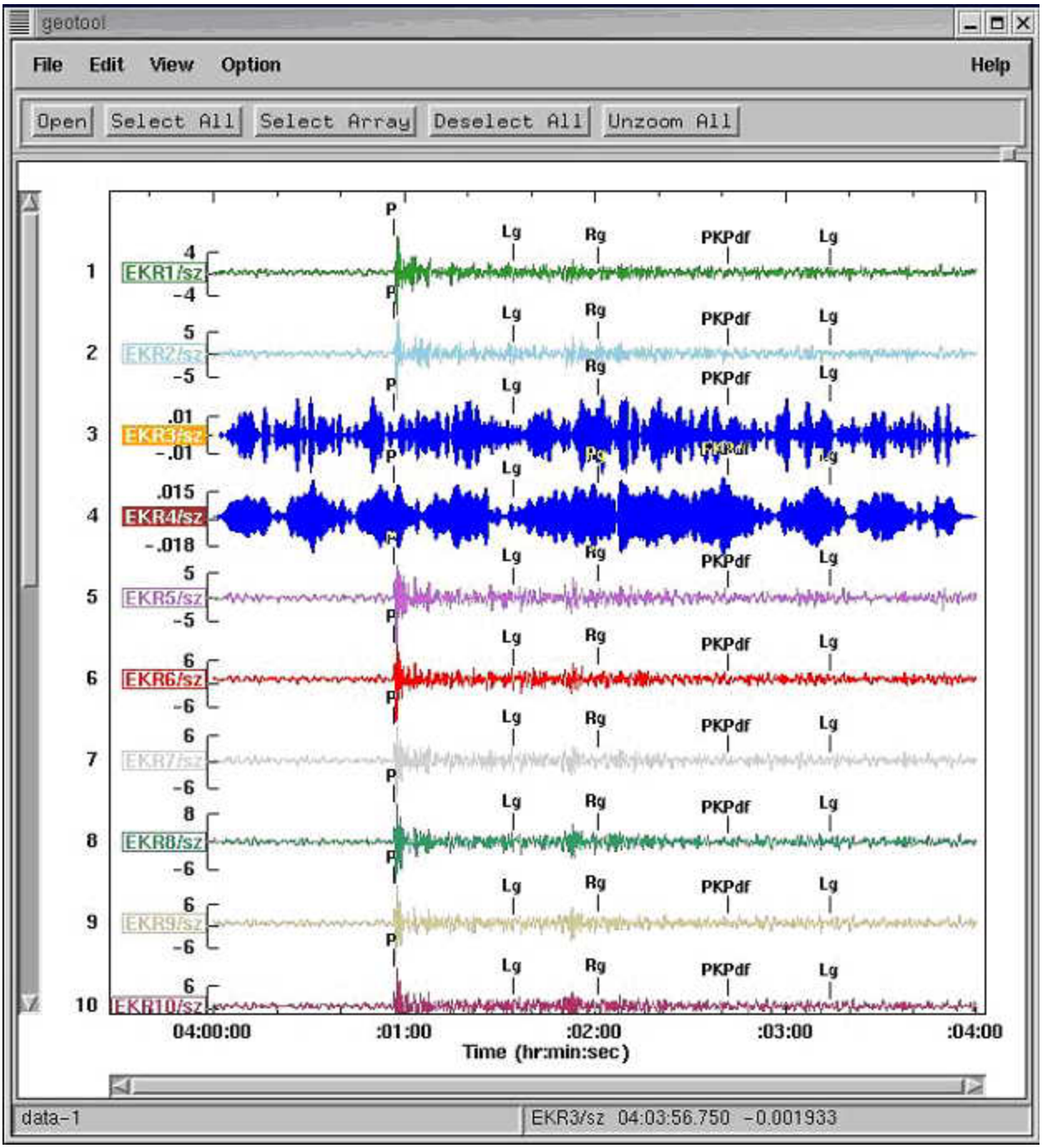

Figure 2. Screen shot of the C-language program *geotool* running on a Linux computer and displaying ten short-period vertical channels recorded by the seismic array EKA. The waveforms on channels 3 and 4 have been passed through a very high-pass filter by a Butterworth-filtering function in *geotool*. These two seismograms are therefore different in this screen image, and in the Linux computer's active memory, from the original versions that still reside on the hard disk. These two altered seismograms and the unaltered ones on channels 1, 6, and 8, along with the *.wfdisc* data structures that describe the seismograms, will be transmitted via CORBA to the Java-language program WaveformViewer running on a Windows 2000 computer. Rather than the disk-resident version of these waveforms, it is the memory-resident version, including the changes made within *geotool* by applying the high-pass filter, which will be transmitted from *geotool* to WaveformViewer.

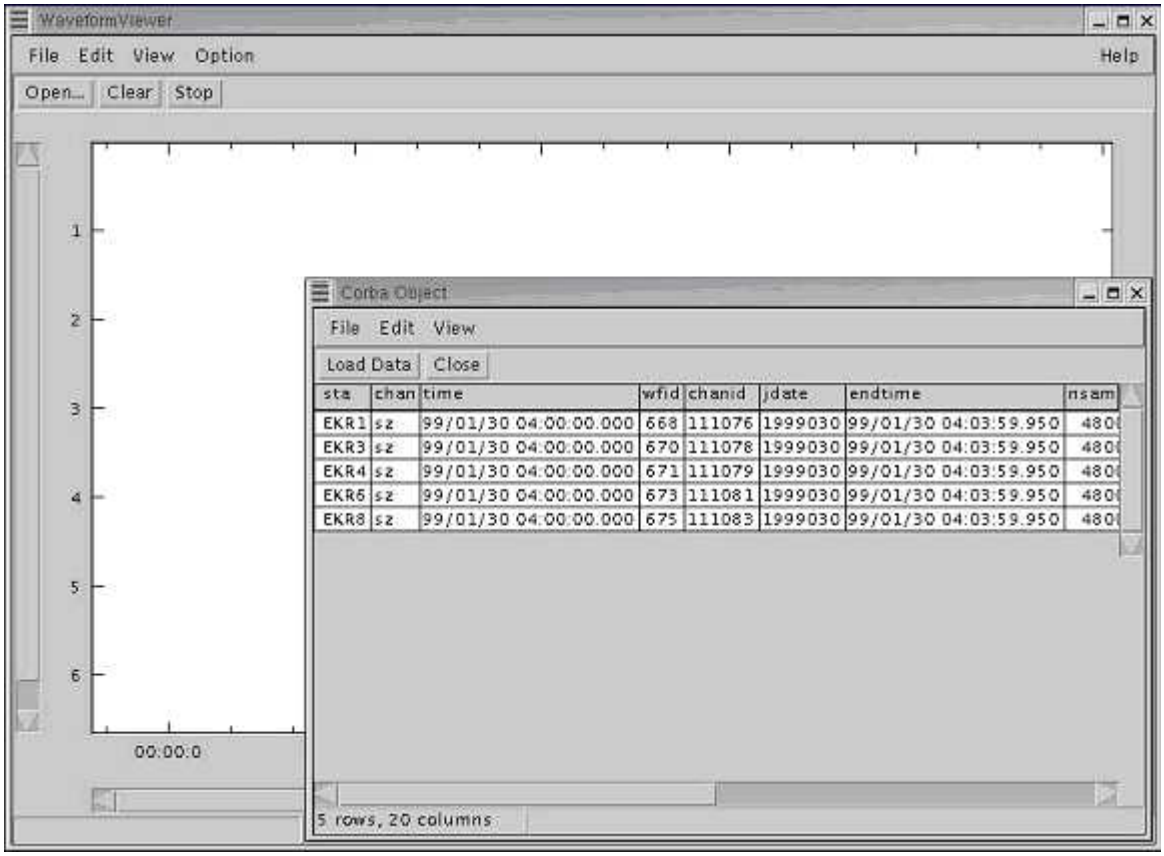

Figure 3. On the Windows 2000 computer running the Java-language program WaveformViewer, the CORBA connection causes a window to pop up that displays the *.wfdisc* record that was sent across from *geotool* on the Linux computer. From that pop-up window, the user selects which waveforms to display. We shall choose to display all five channels that were exported from *geotool*.

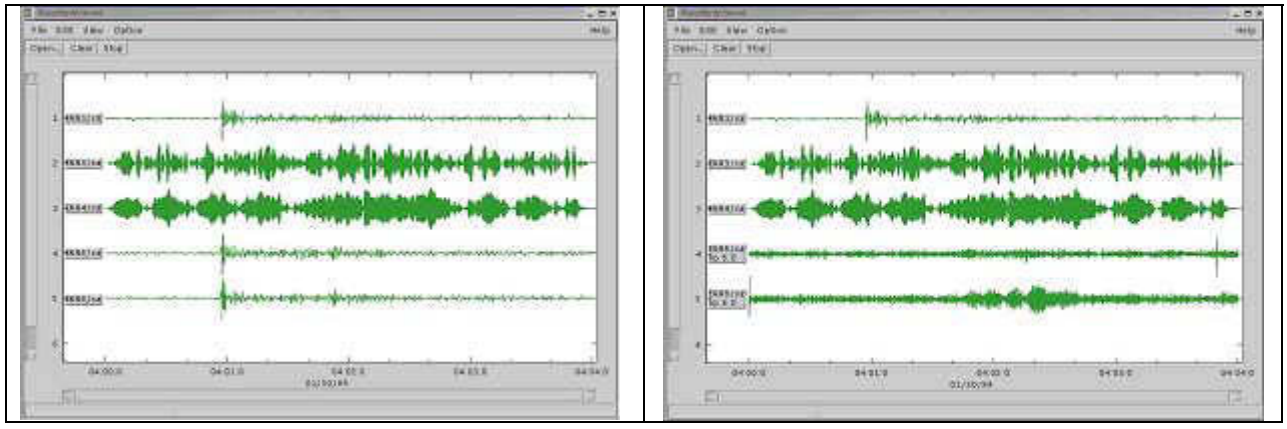

Figure 4. (a) Waveforms imported into WaveformViewer from *geotool*. Note that two of the waveforms reflect the filtering applied by *geotool* and thus show that they have been sent to the Windows computer directly from memory on the Linux computer and not from the disk files. (b) The bottom two waveforms in the display (channels 6 and 8) have now been filtered with WaveformViewer's own Butterworth high-pass filtering routine. A different bandpass was selected in WaveformViewer than in *geotool*, so we can recognize which waveforms were filtered by which program. The waveforms filtered by the WaveformViewer client program will now be sent back from Windows to Linux via the CORBA connection so that they can be displayed within the *geotool* server program.

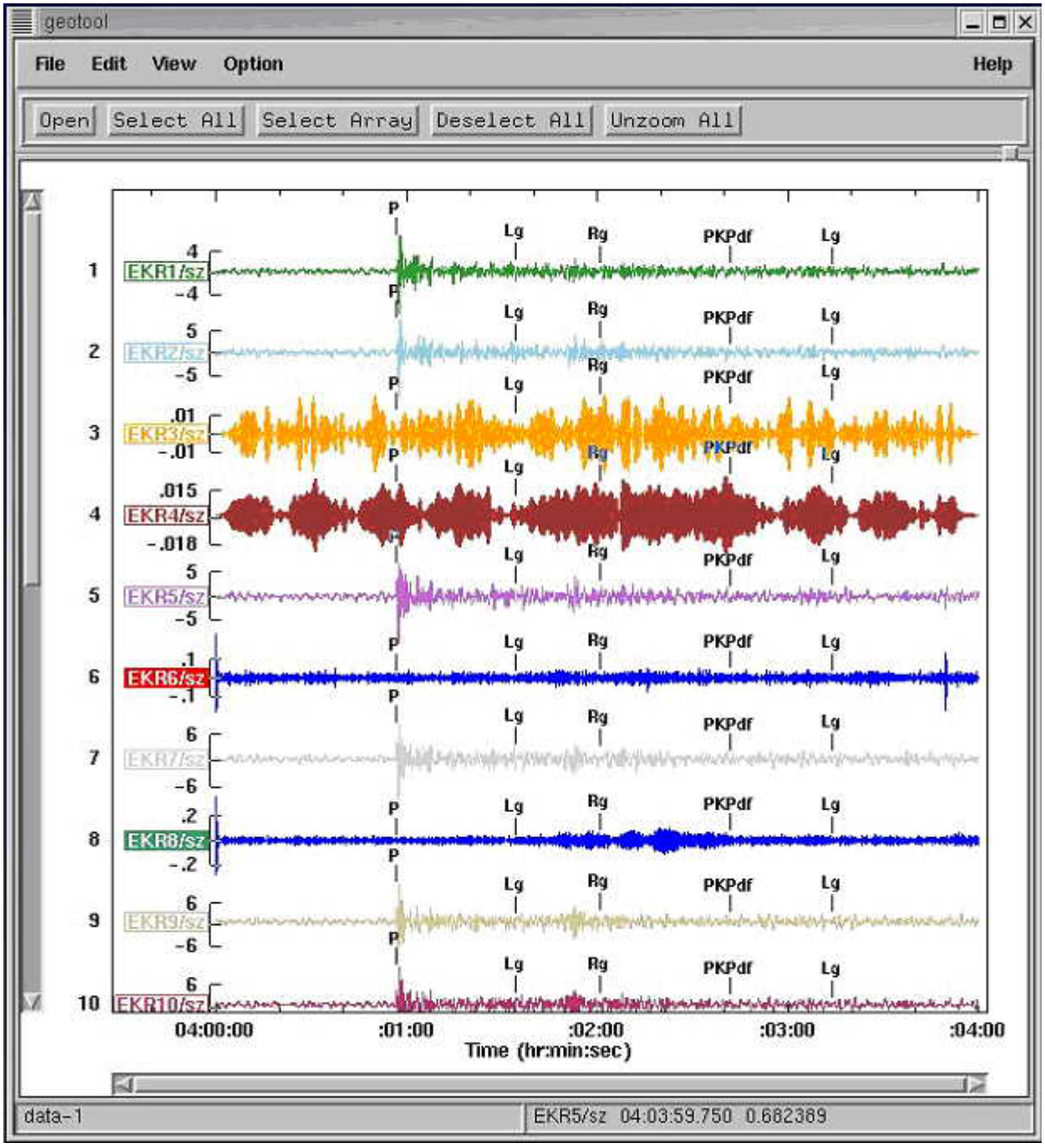

Figure 5. The waveform display that has been running within the *geotool* server program running under Linux is now updated with the seismograms for channels 6 and 8 that were filtered by the WaveformViewer client program running under Windows 2000. Just as with the data transmission from the server to the client, the data that were transmitted back were sent dynamically from the computer's memory rather than being written out as disk files. The whole process shows how a filter could be applied in the reference event analysis system: the data analyst can apply either a filter using a routine that is part of *geotool* or a filter that is part of a separate application (in this case, one written in Java and running on a different computer under a different operating system). This demonstrates the technical feasibility of constructing a reference event analysis system by using CORBA to link stand-alone seismic data analysis programs that are distributed across a network.